BASIL's RIGHT-TYPER; or WRITE YOUR OWN WORD-PROCESSOR PROGRAM

Basil Wentworth
1413 Elliston Drive
Bloomington IN 47401

Over the months that I have been preparing the text of BASIL's COMPENDIUM for publication, I have evolved a word processor program that meets my needs pretty well. The following is a step-by-step description of how I wrote the program, and why I wrote it as I did. The program as described applies to the TS 2068. Some of the principles described can be modified to work on the good ol' ZX81/TS 1000, but only with serious limitations. The biggest problem with the ZX81 is the limited character font, including the absence of lower-case letters. And the membrane keyboard sort of discourages one from thinking in terms of word-processing, unless a keyboard enhancer has been added.

THE AIMS

I had the following aims in mind, listed in order of importance.

1. Most of all, operation of the program should be logical--mimicking the operation of the computer itself, as far as possible.

2. Operation should be as fast as the typist can press the keys. This requirement applies to both ENTER and EDIT modes, and whether the editing is being done at the time the text is entered, or later on.

3. Provision should be made for PRINTing the text on the screen in easy-to-read form; and for LPRINTing hard copy that looks like neatly typed pages. I didn't bother with justifying the right-hand margin (making all lines end in a vertical column).

4. It should be possible to DELETE or MOVE chunks of text.

5. A provision should be included to SAVE the text only, so that the program framework would need to be loaded only once during a session, and the SAVEs of each page of text could be loaded in one by one. In this way, I could economize on tape and time. This is one of the features that I wasn't able to implement on the TS 1000.

6. A character counter (or better, a word counter) would be useful, so the writer can tell how much text he has written.

7. It would have been nice to make the program immune to RUN and CLEAR, but that was a little too complicated. In concept, it could have been done by starting with

```
1 REM                    (thousands of blanks)
```

and then POKEing those blanks with the characters of the text. However, this approach was pretty cumbersome. So I accepted the limitation that I must avoid CLEAR and RUN, using GO TO...instead. It's not too onerous--I just have to stay awake, that's all.

THE ROAD NOT TAKEN (with apologies to Robert Frost)

The ZX81/TS 1000 manual (pge 92 in my copy) gives the following approach:

```
10 LET A$=""
20 IF INKEY$<>"" THEN GO TO 20
30 IF INKEY$="" THEN GO TO 30
40 PRINT INKEY$;
50 LET A$=A$+INKEY$
60 GO TO 20
```

I experimented with this technique, but the "feel" of the keyboard wasn't right, and I didn't see any neat way to provide for editing. So I chose to use the regular facilities that the computer provides for entering data.

THE OVERALL PLAN

The text is held in a string, identified as A$. A CLEAR function initially sets A$ equal to CHR$ 128 (a blank), wiping out any existing information. This is a character that will never be used in any text, so it will serve as an unambiguous indicator of the beginning of the string. Text is then typed into the computer in the regular way, and is successively added to A$ every time the "ENTER" key is pressed.

Ten lines (320 characters) of A$--or all of A$, if there are less than 10 lines--are displayed on the screen. To aid in EDITing, a cursor appears amid the text, to show where the action will take place. The rest of the screen is taken up by a table of instructions showing the effects that various commands will have.

The cursor can be moved by shifted 5-8 keys, while unshifted 0-9 move the 10-line "window" of text displayed (0 moves it backward). Shifted 0 deletes the character at the cursor, and shifted 1 returns you to the MENU. All very much like the normal operation of the computer.

When "ENTER" is pressed, the computer prepares to accept input. Just type in your material as you ordinarily would, and press "ENTER" again to add it to A$ at the point where the cursor is. One CAUTION: whenever you use quotation marks, use doubles ("" for "), just as you would in any string.

Further provisions that take care of PRINTing, LPRINTing, SAVEing, and LOADing will be discussed in due course. But let's start writing the actual program now.

THE PROGRAM

Program lines will be given here in the order in which they are to be entered, rather than in the order in which a LISTing would display them. So don't worry if the order of the line numbers seems a little erratic. Also, don't worry if the finished program has some gaps in the series of line numbers. My copy of the program includes some stuff that's applicable only to my printer, and I'm not going to burden you with it.

And, before we start, a bit of avuncular advice. Whenever you write a long program, it's wise to keep a record of all the variables--both string and ordinary--that you use, so you won't inadvertently use the same one twice. So set aside a piece of paper--or a page in a notebook, if you want to keep a permanent record--and write down the name of each variable at the time you assign it, along with a note on what the variable means or does.

And one more note on programming style. You'll notice that the subroutines are located near the beginning of the program. Since the computer always starts at the head of the program when it searches for a subroutine, you can save a few microseconds each time by putting the most-used subroutines near line 1. It probably won't make any detectable difference in a program of this size, but good habits are worth developing. So now to the program.

SAVE

If you are wise, you'll SAVE your material after every handful of lines, so you won't have so much reconstruction to do after an accidental wipe-out. In a program like this, I make a series of SAVEs on the same tape, so that the first 20 lines, say, would begin at tape counter number 1, the next 20 (plus the first 20 again) would start at number 8, and so on. When both sides of the tape are full, I start at the beginning again, over-writing whatever material was there.

So here's the SAVE routine. Don't worry about the SA's--we'll need them later. Oh, and one last digression. I set off the REMs at the beginning of a section thus:

```
9000 INVERSE-VIDEO REM TRUE-VIDEO SAVE
```

The use of inverse video makes it easy to locate sections when you are analyzing or troubleshooting a program. But that's just a matter of style. You can omit this feature (or even omit the whole line) if you want to. Here we go at last.

```
9000 REM SAVE
9005 LET SA=9
9010 CLS : PRINT AT 3,6; BRIGHT
1;"program"
9015 PRINT AT 3,0;"SAVE"''"Under
 what name do you want to    SAV
E?"
9020 INPUT N$
9030 PRINT ''"Is """;N$;""" OK?"
```

```
9045 GO SUB 45
9050 IF CO THEN GO TO 9100
9055 IF SA>8 THEN GO TO 9000
9100 CLS : PRINT "SAVE """;N$;""
""''"Remove earplug from recorde
r."
9120 IF SA>8 THEN SAVE N$ LINE 1
9135 BEEP .3,12: BEEP 1,12
9140 CLS : PRINT ''"To VERIFY, r
ewind tape, replace earplug, and
 play tape."
9150 PRINT ''"If verification fa
ils, GO TO 1   and try again"
9165 IF SA>8 THEN VERIFY N$
9170 PRINT ' BRIGHT 1;"Verifies
OK"
9175 BEEP .3,12: BEEP 1,12
```

and the appropriate subroutines:

```
  45 REM SUB CONFIRM
  50 IF INKEY$<>"" THEN GO TO 50
  55 PRINT ''"To CONFIRM, press
""C""": GO SUB 20
  60 LET CO=(P=67)+(P=99): RETUR
N
```

and its subroutine:

```
  20 REM SUB INKEY$
  30 IF INKEY$="" THEN GO TO 30
  35 LET P=CODE INKEY$
  40 RETURN
```

And now, a couple of words of explanation. The purpose of the CONFIRM section is to let you review a decision you have made before you ask the computer to execute it. This minimizes the consequences of pressing the wrong button at a critical moment.

Now look at line 60. This line will set CO=1 if you have pressed "C" or "c", and 0 if you have pressed anything else. To help you to understand what goes on here, try PRINT (1=2). The computer should PRINT a 0. Then try PRINT (8>3). This should give 1. Experiment with other expressions in the parentheses. The return should be 1 whenever the expression is true, and 0 when it is false.

And line 9050. This is a sort of shorthand. "IF CO" means "IF CO<>0". Conversely, "IF NOT CO" means "IF CO=0". So line 9050 will send you back to 9100 if you have pressed some key other than "C", and you can try again.

Now it's time to test what you have. And you test it by using it. First, GO TO 9000. The computer should indicate that you're SAVEing the program, and should ask you what name you want to SAVE it under. Enter some name, like "GARBAGE", and see that the computer asks you whether you really meant what you said. You didn't, of course, so press some button other than "C", and give yourself another chance. Since this is your first SAVE, it would be logical to use "1" as the title for it, but you might prefer something like the Greek ALPHA or the Hebrew ALEF. Whatever you want. When the SAVE is finished, you should hear a beeped TA-DA! Now follow instructions to VERIFY. A successful verification should notify you visually and audibly. Now to the next step.

MENU

For the moment, the MENU will be mostly blank. Let's enter the heading and the one item that we do have:

```
 600 REM MENU
 605 CLS
 610 PRINT '"FOR:","PRESS:"
 655 PRINT "SAVE ";BRIGHT 1;"pr
ogram"; BRIGHT 0,9
```

and, to keep the MENU lines in the proper places, enter temporarily:

```
 615 PRINT
 620 PRINT
 625 PRINT
 630 PRINT
 635 PRINT
 640 PRINT
 645 PRINT
 650 PRINT
```

and

```
  10 GO TO 600
```

and to keep you from getting into the SAVE routine without meaning to,

```
8999 STOP
```

Now a GO TO 1 will give you the MENU, which looks like FIGURE 1 at this time.

CLEAR

Before we enter new text, we want to make sure that we have a clean page to write on. Plus which we have to set a few variables so the computer will know what to do. So add to the MENU:

```
 615 PRINT '"CLEAR","CLEAR"
 700 GO SUB 20
 705 LET CO=0
 710 IF P>48 AND P<58 THEN GO TO
 1000*VAL INKEY$
 890 IF P<>88 AND P<>120 THEN GO
 TO 600
 895 CLS : PRINT '"CLEAR- ": GO
SUB 50
 900 GO TO 600+350*CO
 945 REM CLEAR
 950 LET A$=CHR$ 128
 960 LET N1=1
 970 LET CU=1
```

And temporarily put STOPs in as lines 1000, 2000, 3000...etc, up to 8000. Now another test.

GO TO 1 for the new MENU. Press the CLEAR button (letter "X") and you will be asked to CONFIRM. Press something other than "C", and you should get the MENU again. Press "C", and you should get a STOP at 1000. Now verify that N1=1 and CU=1. PRINT A$ will give you an apparent nothing, but PRINT LEN A$ will give you 1, indicating that A$ contains one invisible character. PRINT CODE A$ should give 128. GO TO 1 again. Pressing 0 or any character that is not a number should return you to the MENU. Pressing a number from 1 to 8 should give you a STOP at line 1000, 2000, etc.

ENTER/EDIT

These two functions will share many lines, and they fit together logically from the writer's point of view. Add lines:

```
 620 PRINT "ENTER/EDIT",1
1000 REM ENTER/EDIT
```

And now, the cursor. Start out with

```
   1 LET Z$="xxLxx"
```

Prepare to enter the following, without a line number, but don't

execute it yet.

```
POKE 26719,18: POKE 26720,1: POKE 26722,18: POKE 26723, 0
```

Proofread that line very carefully, cross your fingers, and press "ENTER". Now LIST 1. Line 1 should be a blinking letter "L", just like the one the computer uses to tell you to enter text. Bring that line down into the EDIT position, and add "005" to the line number, so that it now reads "1005". Press "ENTER" to establish it as line 1005. But don't delete LINE 1. We're going to need it later.

```
1015 LET L$="Press ""ENTER"" to
add text."
1020 LET M0=1
1045 GO SUB 65
1050 CLS : PRINT 'A$(N1 TO CU);Z
$;A$(CU+1 TO N2)

  65 REM GO SUB LIMITS
  70 IF N1<1 THEN LET N1=1
  75 IF N1>LEN A$ THEN LET N1=LE
N A$
  80 LET N2=N1+318
  85 IF N2>LEN A$ THEN LET N2=LE
N A$
  90 IF CU<N1-1 THEN LET CU=N1-1
  95 IF CU<1 THEN LET CU=1
 100 IF CU>N2 THEN LET CU=N2
 110 RETURN
```

Time for another test. GO TO 1, CLEAR and CONFIRM. The program should STOP at line 2000. Enter

```
LET A$=CHR$ 128+"Basil loves Jocelyn."
```

Now GO TO 1 and press "1". You should see this text displayed at the top of the screen, along with the blinking cursor. Unfortunately, you can't do anything with it yet, though. Continue with

```
1055 GO SUB 120

 120 REM GOSUB LEGENDS
 125 PRINT AT 13,0;LEN A$-1,"LIN
E NO. ";1+INT (N1/32)
 130 PRINT '"1-9 advances line;
0 back 10"
 135 PRINT "Shifted 5-8 MOVES cu
rsor"
 140 PRINT "Shifted 0 DELETES ch
aracter"
 145 PRINT "Shifted 1 gives MENU
"
```

```
150 PRINT L$
155 RETURN
```

Another test. CLEAR, and enter a longer text this time. Something like

```
LET A$=CHR$128+"Now is the time for all good persons to come to
the aid of their pLLIST rties."
```

GO TO 1 and pPAUSE ess "1". The screen should display that "window" of text, looking like FIGURE 2. Let's work with that window.

```
1065 REM MOVE WINDOW
1070 GO SUB 20
1075 IF P=7 THEN GO TO 600
1110 IF P<48 OR P>57 THEN GO TO
1150
1125 LET N1=N1+32*VAL INKEY$-320
*(P=48)
1130 LET CU=CU+32*VAL INKEY$-320
*(P=48)
1135 GO TO 1030
```

Display the window as you did before, and experiment with pressing 1, 2, and 0, to see the window move up and down through the limited space that the short length of the text permits. Notice that the line number displayed at the bottom of the text changes as the window moves. This feature is useful during proofreading. Now to get that cursor to move.

```
1150 REM MOVE CURSOR
1155 IF (P<8 OR P>11) AND P<>37
AND P<>40 THEN GO TO 1200
1160 LET CU=CU+(P=9)-(P=8)+5*((P
=40)-(P=37))+32*((P=10)-(P=11))
1165 IF CU<N1 THEN LET N1=CU
1170 IF CU>N2 THEN LET N1=CU-318
1180 GO TO 1030
```

You should be able to move the cursor through the text--up, down, and sideways. Pressing 5 or 8 with SYMBOL SHIFT moves the cursor 5 spaces. Now to get some control over the text itself.

```
1200 REM ADD TEXT
1205 IF P<>13 AND P<>42 THEN GO
TO 1250
1215 IF MO<>1 THEN RETURN
1220 INPUT T$
1225 IF T$="" THEN LET T$=""
1235 LET A$=A$( TO CU)+T$+A$(CU+
1 TO)
1240 LET CU=CU+LEN T$: LET N2=N2
+LEN T$: LET N1=N2-318
1245 GO TO 1000
```

Now experiment a bit. GO TO 1, CLEAR, and then press "ENTER". The computer should show the "" prompt at the bottom of the screen, indicating that it expects a string input. You can get rid of that prompt by pressing "ENTER" again. Or you can type in some text and then press "ENTER". And, by the way, you can always return to the MENU by pressing shifted "1" at a time when the prompt is absent. Any text that you type in and ENTER should appear on the screen, at the point at which the cursor was located.

That blob defined by line 1225 is used to indicate the end of a line. This particular symbol was chosen because it has a strongly identified vertical stroke. It is much more satisfactory to show two line feeds by "" than it would be by "", for instance.

Now we lack only one function to complete our EDIT facility:

```
1250 REM DELETE
1255 IF P<>12 THEN GO TO 1030
1260 LET A$=A$( TO CU-1)+A$(CU+1
 TO)
1270 LET CU=CU-1
1280 GO TO 1030
```

And, while you're at it,

```
2000 GO TO 600
```

There you have the basics of ENTERing and EDITing text. However, your prose isn't exactly impressive, looking something like FIGURE 3. We have to provide a way to PRINT (and LPRINT) it.

PRINT

See pp 20-21 for m/c alternative

Here I have to apologize for taking the easy way out. If there ever was a situation that cries out for machine code, this PRINT sequence is it. It would only take a couple dozen bytes to do the job, and BASIC proceeds at a frustratingly leisurely pace. However, I've just been too lazy to translate the program into machine code. Maybe some of you computer mavens will beat me to it.

The delay is caused mainly by the time the computer takes to make sure that each line finishes at the end of a word. I debated giving it the option of ending a line at a hyphen as well, but abandoned that approach for two reasons. First, I knew that I would be asking for trouble whenever I used two hyphens together to indicate a dash. The other reason is that it's hard to keep track, when you're writing text into one long string, of where the hyphens will have to go. So the hyphens had to go, in a manner of speaking.

So you could get
supercallifragilisticexpialidocious, or something of the sort, where a line will be ridiculously short just because the computer has to get a deep breath to print a sesquipedalian word that's coming up.

Here, in a nutshell, is how the PRINT section works. First, you tell the computer how long a line you want (up to 96 characters--you could adapt the program to accommodate more, if you wanted to). You also specify what length of line the printer is set for. If you are printing to the screen, or using the TS 2040 printer, then the latter number will be 32. If you ask for a line too long for the printer to handle in one go, the program will print first the left-hand side of the text, and then the right-hand side, like this:

```
Now is the time for all good per
parties.


sons to come to the aid of their
```

and you can cut them out and paste them together, side by side. Usually, though, if you are using a printer with full-length line capacity, your printed line will be equal to or less than that capacity. Unless you're printing wallpaper.

Next you tell the computer whether you want hard copy, and whether you want the print to start at the beginning of the text or at the top of the "window". The latter choice is useful when you are proofreading.

After you have indicated (and confirmed) these choices, the computer gets to work. It takes a chomp out of A$ equal to the length of line you have specified, and calls this mouthful X$. Then it looks through X$ for a LINE FEED instruction. If it finds one, it cuts off X$ right there, filling in the rest of the length with blanks. And now it PRINTs (or LPRINTs) enough of X$ to fill up the printer's capacity. Or all of X$, if the printer can handle it.

If there is no LINE FEED instruction, the computer looks at the last character in X$. If this is not a blank, then it chops off the last character, and looks again. And again, and again... And that's what takes the time. When X$ has been chopped back to where it ends in a blank, then the line is PRINTed or LPRINTed as described above. Meanwhile, the computer has kept track of the numbers so that the next bytes of A$ it looks at will fit into the scheme.

And so on to the end of A$, and then repeating to produce the second side of X$ if necessary. And the third. When the computer gets to the end, it gives the BEEPed TA-DA! so you can be writing limericks or reading your Perry Mason book while the printer works its far-too-leisurely way through the text.

Here are the program lines that take care of the PRINT and LPRINT function:

```
8000 REM PRINT
8010 IF A$(LEN A$)<>" " THEN LET
 A$=A$+" "
8020 LET LW=32: REM No of CHRs w
anted per line>>>not to esceed 9
6
8030 LET LP=32: REM Line capacit
y of printer (or screen)
8040 CLS :  PRINT ''"CHRs per li
ne set at ";LW'" by line 8020"
8120 PRINT ''"Line capacity of p
rinter (or   screen) set at ";L
P;'" by line 8030"
8125 PRINT '' BRIGHT 1;"OK?"
8135 GO SUB 45
8145 IF NOT CO THEN CLS : LIST 8
020
8200 CLS : PRINT ''"Hard copy?"'
'"  Y-  N-"
8205 IF INKEY$<>"" THEN GO TO 82
05
8210 GO SUB 20
```

```
8215 PRINT ''"You have chosen "
;BRIGHT 1;CHR$ P
8220 LET HA=(P=89)+(P=121)
8225 GO SUB 45
8230 IF NOT CO THEN GO TO 8200
8240 CLS : PRINT ''"Start at beg
inning?   (B- )"
8245 PRINT ''"   or resume?    (
R- )"
8250 IF INKEY$<>"" THEN GO TO 82
50
8255 GO SUB 20
8260 LET YE=(P=66)+(P=98)
8265 PRINT ''"You have chosen  "
; BRIGHT 1;CHR$ P
8270 GO SUB 45
8275 IF NOT CO THEN GO TO 8240
8280 IF YE THEN LET ST=2
8285 IF NOT YE THEN LET ST=N1
8290 IF HA THEN PRINT '' FLASH 1
;"Turn on"; FLASH 0;" printer an
d SEL button,"'"       and press
 ""ENTER""": INPUT Z$
```

and

```
650 PRINT "PRINT",8
```

Now some more tests. GO TO 1 and press "8". Follow through with various answers to the questions asked, and see whether the computer gives the logical response in each case. Check to see that HA is 1 when you ask for hard copy, and 0 when you don't. That ST=2 if you ask to start the print at the beginning, and that ST=N1 when you ask to RESUME printing.

Now to the PRINTing sequence itself.

```
8500 CLS
8510 LET R=INT ((LW-1)/LP)+1
8530 FOR F=1 TO R
8540 LET A=ST
8550 LET SP=A+LW-1
8600 IF SP>LEN A$-1 THEN LET SP=
LEN A$-1
8610 LET X$=""
8620 FOR Z=A TO SP
8630 IF A$(Z)<>"" THEN LET X$=X
$+A$(Z): NEXT Z
8640 IF A$(Z)="" THEN LET Z=Z+1
: GO TO 8710
```

```
8650 IF A$(Z)=" " OR Z>=LEN A$ T
HEN GO TO 8710
```

```
8700 LET X$=X$( TO LEN X$-1): LE
T Z=Z-1: GO TO 8650
8710 IF A=1 THEN GO TO 8740
8720 IF A$(A-1)="" THEN GO TO 8
740
8730 IF X$(1)=" " THEN LET X$=X$
(2 TO )
8740 LET A=Z
8750 LET X$=(X$+"    (96 spaces)
     ")( TO LW)
8800 LET P1=(F-1)*LP+1
8810 LET P2=P1+LP-1
8820 IF P2>LW THEN LET P2=LW
8830 IF HA THEN LPRINT X$(P1 TO
P2)
8840 IF NOT HA THEN PRINT X$(P1
TO P2)
8850 IF A<LEN A$ THEN GO TO 8550
8910 NEXT F
8920 BEEP .3,12: BEEP 1,12
8999 STOP
```

And that's the biggest part of it. The rest are frills. But rather convenient frills to have. For instance, to help you proofread your text as you write it:

PROOFREAD

```
 635 PRINT "PROOFREAD",4

4000 REM PROOFREAD
4005 CLS
4010 LET LW=32: LET LP=32
4020 LET HA=0: LET DS=0: LET ST=
N1
4025 IF A$(LEN A$)<>" " THEN LET
 A$=A$+" "
4030 GO TO 8510
```

Now, when you're entering text, the single MENU item "4" will PRINT on the screen the text that is in the window (and beyond), all arranged in neat lines, without text wraparound. You're in trouble, though, if you have any of those Disney-type words that have more than 32 letters in them. Sorry about that.

DELETE/MOVE

This is another feature that can save time in EDITing. Sometimes you might like to DELETE or MOVE large blocks of text,

and this method is less cumbersome than the character-by-character procedure.

```
 630 PRINT "DELETE/MOVE",3

1100 IF P=13 AND MO=3 THEN LET A
$=A$( TO CU)+">"+A$(CU+1 TO)
1105 IF P=13 AND MO=4 THEN LET A
$=A$( TO CU)+"<"+A$(CU+1 TO)

3000 REM DELETE/MOVE
3005 LET MO=3
```

Now, to get a cursor, make sure that line 1 is still intact, and POKE 26721,62. Line 1 should now show blinking ">". Change this line number to 3010, but leave line 1 for future use.

```
3020 LET L$="Press ""ENTER"" to
mark START of      passage to be
moved"
3030 GO SUB 1030
3040 LET D1=CU
```

Make sure that line 1 is still intact, and POKE 26721,60. You should now have a blinking "<". Bring it down to line 3100, but again leave line 1 intact.

```
3105 LET MO=4
3110 LET L$="Press ""ENTER"" to
mark END of         passage to be
moved"
3120 GO SUB 1030
3130 LET D2=CU
3140 IF D1>D2 THEN CLS : PRINT '
' BRIGHT 1;"You have asked for t
he end to bebefore the start.  T
ry again.    ": go to 610
```

Again another cursor. This time, after checking that line 1 is there, POKE 26721,94. Move it to 3150.

```
3155 LET MO=5
3160 LET L$="Press ""ENTER"" to
mark point of      insertion; ""*
"" to DELETE."
3170 GO SUB 1030
3180 LET D3=CU
3195 REM DELETE
3200 IF P=13 THEN GO TO 3300
```

```
3210 LET ND1=D1-250: IF ND1<1 TH
EN LET ND1=1
3215 LET ND2=D2+250: IF ND2>LEN
A$ THEN LET ND2=LEN A$
3220 CLS : PRINT A$(ND1 TO D1)+A
$(D2+2 TO ND2)
3230 GO SUB 50
3235 IF NOT CO THEN GO TO 3000
3240 LET A$=A$( TO D1)+A$(D2+2 T
O )
3250 GO TO 1000
3299 REM MOVE TEXT BACKWARD
3300 IF D3>D2 THEN GO TO 3400
3310 LET ND1=D3-250: IF ND1<1 TH
EN LET ND1=1
3315 LET ND2=D2+250: IF ND2>LEN
A$ THEN LET ND2=LEN A$
3320 CLS : PRINT A$(ND1 TO D3)+A
$(D1+2 TO D2)+A$(D3+1 TO D1)+A$(
D2+2 TO ND2)
3330 GO SUB 50
3335 IF NOT CO THEN GO TO 3000
3340 LET A$=A$( TO D3)+A$(D1+2 T
O D2)+A$(D3+1 TO D1)+A$(D2+2 TO
)
3350 GO TO 1000
3405 REM MOVE TEXT FORWARD
3410 LET ND1=D3-250: IF ND1<1 TH
EN LET ND1=1
3415 LET ND2=D2+250: IF ND2>LEN
A$ THEN LET ND2=LEN A$
3420 CLS : PRINT A$(ND1 TO D1)+A
$(D2+2 TO D3)+A$(D1+2 TO D2)+A$(
D3+1 TO ND2)
3430 GO SUB 50
3435 IF NOT CO THEN GO TO 3000
3440 LET A$=A$( TO D1)+A$(D2+2 T
O D3)+A$(D1+2 TO D2)+A$(D3+1 TO
)
3450 GO TO 1000
```

Test these routines. GO TO 1, and press "3". The cursor will now appear as a flashing ">". Position this symbol , just as you would the ordinary cursor, until it points to the first character of the text that you want to move (or remove). When you have it where you want it, freeze it there by pressing "ENTER".

A new cursor now appears--a flashing "<", while the old cursor, without its flash, stays where you froze it. Move the new one to the end of the text you want to (re)move, and freeze it in place by pressing "ENTER". A new flashing cursor now appears (this begins to sound like the story of Hercules and the Hydra)--this time an up-pointing arrow. Meanwhile, the second one, now flashless, remains to complete the outline of the text to be moved. If you want to DELETE this text, press "*". If you want to move it, maneuver the arrow to place where you want to insert it; then press "ENTER" once more.

In either case, the screen will display a segment of the text, showing the changes you have proposed. If you accept the changes, CONFIRM by pressing "C"; if not, press any other key. If you choose to renege on your change, you'll have to DELETE the frozen left-over cursors.

AND NOW THE DATA SAVE

Page 149 of the TS 2068 manual tells how to SAVE the contents of a string array. Notice that this procedure is a little more discriminating than my grandmother, who used to save any old string. The computer will save a string only if it is in an ARRAY, properly DIMENSIONed and all. And when you LOAD a string that has been SAVED in this way, it comes back as an array.

To see what this means, first SAVE whatever you have in the computer, and then try the following. Use direct commands, with no line numbers.

```
LET A$="BASIL LOVES JOCELYN."

SAVE "JO" DATA A$()
```

"DATA", of course, is the keyword above letter "D".

Now try to get the material back from the tape:

```
LOAD "JO" DATA W$()
```

Well, everything looks fine until you try to PRINT W$. H'm...very strange. The machine tells you that you have the wrong subscript, and you don't remember even asking for a subscript. Apparently, however, you did so implicitly when you asked the computer to deal with what it thought was an array.

Well, nothing daunted, let's do it the computer's way. First, CLEAR the computer. LET A$="Basil loves Jocelyn." And now...

```
DIM W$(LEN A$)
LET W$=A$
SAVE "JO" DATA W$()
```

Now CLEAR, just to make sure the computer isn't cheating on you. Or turn the computer off for a few seconds, just to make really sure. And try LOADing:

```
LOAD "JO" DATA W$()
```

Now PRINT W$ gives you what you expected. And what does PRINT A$ give? Well, since you CLEARed the computer, and didn't SAVE A$, you shouldn't be too surprised that A$ is now unknown. But not to worry. We have W$ to work with. So let's go on:

```
LET W$=W$+"Today."
```

PRINT W$ again. WHAT?! Well, since you DIMensioned off a block of memory just big enough to hold the old W$, you shouldn't feel too outraged when you can't squeeze any more bytes into it. As Dolly Parton said, you can't carry ten pounds of mud in a five-pound sack. So here's what you do.

```
LET A$=W$
```

and you'll find that you can manipulate A$ just as you used to. And now we don't need that W$ any more. Let's retire it. LET W$="" ought to do it. Try it, and then PRINT W$. Good--it's gone. It isn't, you know. PRINT LEN W$. Surprise! It seems that you still have that five-pound sack to deal with, even though it's empty. The best you can do is to fold it up as small as you can:

```
DIM W$(1)
```

We know enough by now to write the SAVE data and LOAD date routines. ReLOAD that program you SAVEd, and continue:

```
 640 PRINT "LOAD "; BRIGHT 1;"d
ata"; BRIGHT 0,5
 645 PRINT'"SAVE "; BRIGHT 1;"d
ata"; BRIGHT 0,7

7000 REM SAVE data
7010 LET SA=7
7020 DIM W$(LEN A$)
7030 LET W$=A$
```

```
7040 CLS : PRINT AT 3,6; BRIGHT
1;"DATA"
7100 GO TO 9015
```

and

```
5000 REM LOAD data
5005 CLS : PRINT '"Loading "; BR
IGHT 1;"data"; BRIGHT 0;''"Start
 tape"
5010 LOAD "" DATA W$()
5015 LET A$=W$
5020 DIM W$(1)
5030 LET N1=1
5040 LET N2=320
5050 IF N2>LEN A$ THEN LET N2=LE
N A$
5060 LET SC=1
5100 GO TO 1
```

```
9060 IF SA<8 THEN GO TO 7000
9130 IF SA<8 THEN SAVE N$ DATA W
$()
9160 IF SA<8 THEN VERIFY N$ DATA
W$()
```

AND FINALLY, THE WORD COUNTER

Here again I should apologize, as this is another job that should be done by machine code. But here's how to do it in BASIC, anyway.

```
9989 STOP
9990 REM WORD COUNT
9991 LET W=0
9992 CLS : PRINT ''FLASH 1;"I'm
thinking! I'm thinking!"
9993 FOR F=1 TO LEN A$
9994 IF A$(F)<>" " THEN NEXT F
9995 IF A$(F-1)=" " THEN NEXT F
9996 LET W=W+1
9997 NEXT F
9998 CLS : PRINT W
9999 BEEP .3,12 : BEEP 1,12
```

To activate it, just GO TO 9990, and find something to keep you busy while it grinds away.

And there you are. It's been a long haul, but I hope that you've found it worth while. And that maybe you've learned

something along the way. I certainly have.

See Listing

```
 950 REM  CLEAR
 960 CLEAR : DIM x$(190)
 970 LET x$(150 TO 186)="*K\ GO SUB J"+CHR$ 6+CHR$ 17+CHR$ 14+"#"++CHR$ 12+
"(>RANDOMIZE "+CHR$ 6+CHR$ 0+CHR$ 13+"<>"
 980 LET a$=CHR$ 128: LET N1=1: LET CU=1: LET PICA=1
 985 LET MA=10: LET TY=4: LET ST=1
 990 LET LL=65: LET LS=1: LET RE=1
1005 LET Z$="L"
1015 LET L$="Press ""ENTER"" to add text."
1020 LET MO=1
1045 GO SUB 65
1050 CLS : PRINT 'A$(N1 TO CU);Z$;A$(CU+1 TO N2)
1055 GO SUB 120
1060 REM ...
1070 REM ...
1080 REM ...
8000 REM   PRINT
8010 CLS : PRINT ''"Hard copy?   Y-  N-     "
8020 GO SUB 20
8030 PRINT AT 2,20; BRIGHT 1;INKEY$: LET HA=(INKEY$="y")+(INKEY$="Y")
8040 GO SUB 50: IF NOT CO THEN  GO TO 8000
8050 LET LL=65
8051 REM line length
8060 CLS : PRINT ''"Do you want high quality (slow) print?       y-    n-"
8070 GO SUB 20
8080 PRINT AT 3,24; BRIGHT 1;INKEY$: LET NLQ=(INKEY$="y")+(INKEY$="Y")
8090 GO SUB 50: IF NOT CO THEN  GO TO 8060
8500 REM   PRINT proper
8510 CLS : PRINT ''"Start at beginning (1)"
8520 PRINT "  or resume (2)?        "; BRIGHT 1;RE
8530 GO SUB 8990
8550 IF Z$<>"" THEN  LET RE=VAL Z$: GO TO 8500
8560 IF NOT HA THEN  LET LL=32: CLS
8600 LET FB=2
8610 IF RE=2 THEN  LET FB=N1
8620 IF A$(LEN A$)<>" " THEN  LET A$=A$+" "
8630 IF LL>LEN A$+1-FB THEN  LET LL=LEN A$+1-FB
8640 LET X$( TO LL)=A$(FB TO )
8650 LET X$(159)=CHR$ LL
8660 LET a=PEEK 23627+256*PEEK 23628
8670 LET B=USR (a+155)
8680 IF NOT HA THEN  PRINT x$( TO B)
8690 IF HA  THEN  LPRINT CHR$ 27;"x";CHR$ (48+nlq);CHR$ 27;"P";CHR$ 27;"N";
CHR$ 12;CHR$ 27;"l";CHR$ 5;x$( TO B)
8695 REM P=pica;NCHR$ .. gives page length;lCHR$ .. give l.h. margin
8700 LET FB=FB+B+1
8710 IF FB<LEN A$ THEN  GO TO 8630
8715 IF ha THEN  LPRINT CHR$ 12
8720 BEEP .3,12: BEEP 1,12: STOP
8990 INPUT "If OK, press ""ENTER""; if not,    input new value  ";Z$
8995 RETURN
```

Per line 970, CODEs of X$(150 TO 186) are:

150 42
151 75
152 92
153 1
154 4
155 0
156 237
157 74
158 6
159 9
160 14
161 0
162 35
163 12
164 5
165 40
166 7
167 62
168 139
169 190
170 40
171 11
172 24
173 244
174 62
175 32
176 190
177 40
178 4
179 13
180 43
181 24
182 249
183 6
184 0
185 13
186 201